**Đại học Bách khoa Hà Nội**

**Trường Công nghệ thông tin và truyền thông**

**-----o0o-----**

****

**BÁO CÁO THỰC HÀNH**

**Môn: Thực hành lập trình hướng đối tượng**

**Học phần: IT3103**

**Mã lớp: 721428**

**LAP 04 : 08/12/2022**

**Giảng viên hướng dẫn: Lê thị Hoa**

**Sinh viên thực hiện: Lê Thế Anh**

**MSSV: 20200018**

**Mục lục**

[1. Import the existing project into the workspace of Eclipse 4](#_Toc121606810)

[2. Additional requirements of AIMS 4](#_Toc121606811)

[3. Creating the **Book** class 5](#_Toc121606812)

[4. Creating the abstract **Media** class 7](#_Toc121606813)

[5. Creating the **CompactDisc** class 9](#_Toc121606814)

[5.1. Create the **Disc** class extending the **Media** class 9](#_Toc121606815)

[5.2. Create the **Track** class which models a track on a compact disc and will store information incuding the **title** and **length** of the track 11](#_Toc121606816)

[5.3. Open the **CompactDisc** class 11](#_Toc121606817)

[6. Create the Playable interface 13](#_Toc121606818)

[7. Update the **Cart** class to work with **Media** 15](#_Toc121606819)

[8. Update the **Store** class to work with **Media** 16](#_Toc121606820)

[9. Constructors of whole classes and parent classes 17](#_Toc121606821)

[10. Unique item in a list 18](#_Toc121606822)

[11. Polymorphism with toString() method 19](#_Toc121606823)

[12. Sort media in the cart 20](#_Toc121606824)

[13. Create a complete console application in the Aims class 22](#_Toc121606825)

# Import the existing project into the workspace of Eclipse

- Open Eclipse

- Open File -> Import. Type zip to find Archive File if you have exported as a zip file before. You may choose Existing Projects into Workspace if you want to open an existing project in your computer. Ignore this step if the AimsProject is already opened in the workspace.

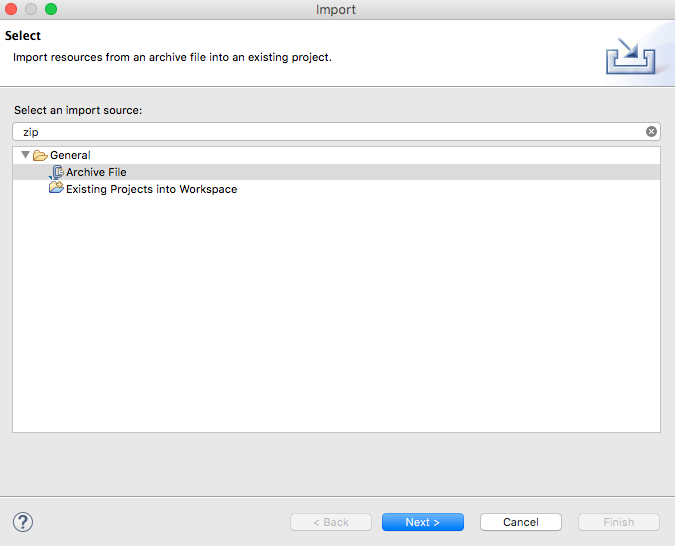
****

Figure 1. Import existing project

- Click Next and Browse to a zip file or a project to open

Once the project is imported, you should see the classes you created in the previous lab, namely, **Aims**, **Cart**, **DigitalVideoDisc**.

**-** We can apply Release Flow here by creating a branch, e.g., **topic/aims-project/add-media-class**, writing our codes, testing them, pushing them, and then merging it with master.

# Additional requirements of AIMS

Starting from this lab, you extend the AIMS system that you created in the previous exercises to allow customer to order 2 new types of media: books and CD.

A book’s information includes: id, title, category, cost and list of authors.

A CD’s information includes: id, title, category, artist, director, track list and price. Additionally, each track is unique in a CD with its own title and length. The length of a CD is sum of the lengths of its tracks.

When a user sees the details of a media in the store, the information displayed depends on the type of media.

* For books, the system shows their title, category, author list, the content length (i.e., the number of tokens).
* For CDs, the system displays the CD’s information (i.e. CD title, category, artist, director, CD length, and the cost for the CD) and then displays the information of all the tracks in that CD.
* For DVDs, the system displays the DVD’s information (i.e. DVD title, category, director, DVD length, and the cost for the DVD).

Additionally, the user can choose to play some media when browsing the list of media in the store or seeing the current cart. For simplicity, we establish the way the system plays a media is as follows: When a CD is played, the system displays the CD information (i.e., CD title and CD length) and plays all the tracks of the CD. To play a track, the system displays the track’s name and its length. Similarly, a DVD can also be played, i.e., the system displays the title and length of the DVD. If a DVD or track has the length 0 or less, the system must notify the user that the track, the DVD or the CD of that track cannot be played.

# Creating the **Book** class

- In the Package Explorer view, right-click the project and select New -> Class. Adhere to the following specifications:

* Package: **hust.soict.dsai.aims.media**
* Name: **Book**
* Access modifier: **public**
* Superclass: **java**.**lang**.**Object**
* **public** **static** **void** **main**(**String**[] **args**): **do not check**
* Constructors from Superclass: **Check**
* All other boxes: **Do not check**

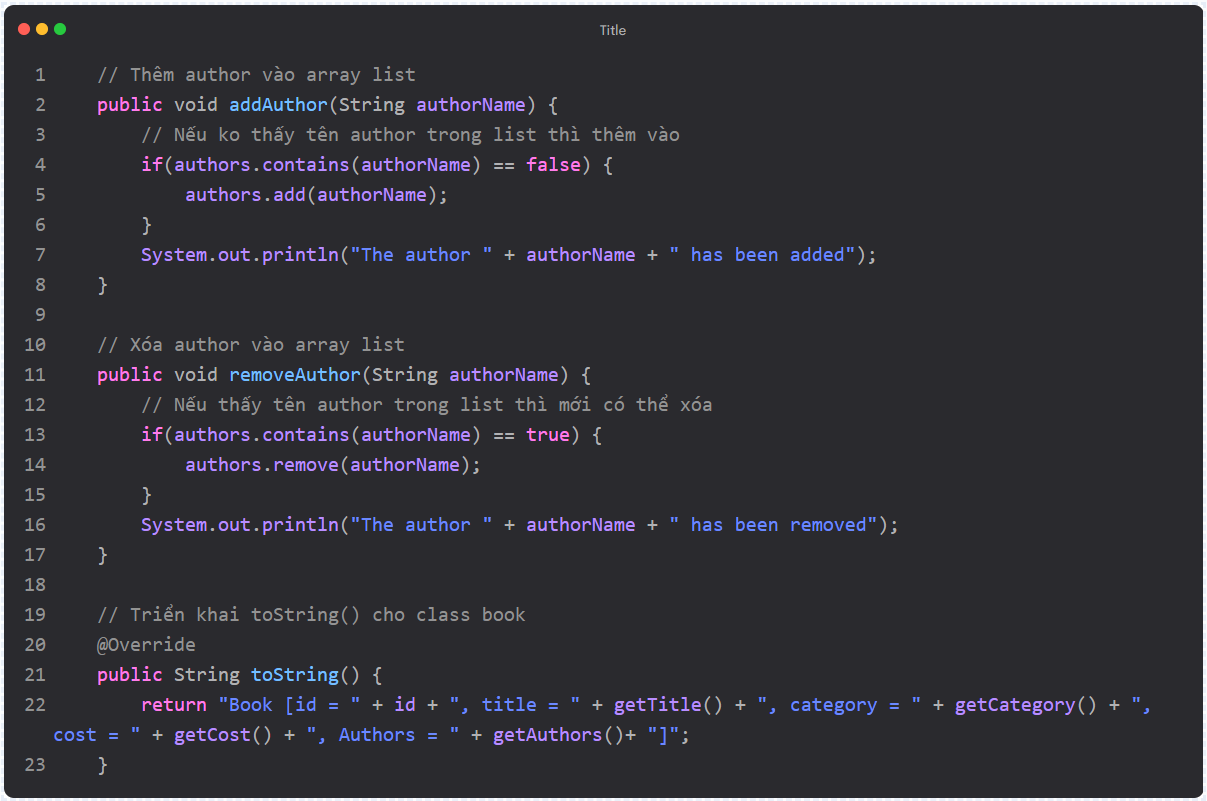
**Add fields to the Book class**

* To store the information about a **Book**, the class requires five fields: an **int** field **id,** **String** fields **title** and **category**, a **float** field **cost** and an **ArrayList** of **authors**. You will want to make these fields private, with public accessor methods for all but the **authors** field.



Figure 2. Adding fields to Book class

* Next, create **addAuthor**(**String** **authorName**) and **removeAuthor**(**String** **authorName**) for the **Book** class
  + The **addAuthor**(...) method should ensure that the author is not already in the **ArrayList** before adding
  + The **removeAuthor**(...) method should ensure that the author is present in the **ArrayList** before removing
  + Reference to some useful methods of the **ArrayList** class



# Creating the abstract **Media** class

At this point, the **DigitalVideoDisc** and the **Book** classes have some fields in common namely id, title, category and cost. Here is a good opportunity to create a common superclass between the two, to eliminate the duplication of code. This process is known as refactoring. You will create an abstract class called **Media** which contains these fields and their associated get and set methods.

**Create the Media class in the project**

- In the **Package Explorer** view, right click to the project and select New -> Class. Adhere to the following specifications for the new class:

* Package: **hust.soict.dsai.aims.media**
* Name: **Media**
* Access Modifier: **public**, **abstract**
* Superclass: **java**.**lang**.**Object**
* Constructors from Superclass: Check
* **public static void main (String[] args)**: do not check
* All other boxes: Do not check

- Add fields to the **Media** class

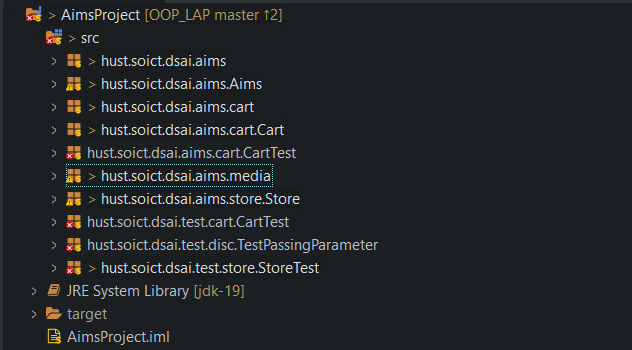
* To store the information common to the **DigitalVideoDisc** and the **Book** classes, the **Media** class requires four private fields: **int id**, **String** **title**, **String** **category** and **float** **cost**
* You will want to make public accessor methods for these fields (by using **Generate Getter and Setter** option in the **Outline** view pop-up menu)

Ảnh có chứa văn bản

Mô tả được tạo tự động

- Remove fields and methods from **Book** and **DigitalVideoDisc** classes

* Open the Book.java in the editor
* Locate the Outline view on the right-hand side
* Select the fields id, title, category, cost and their accessors & mutators (if exist)
* Right click the selection and select Delete from the pop-up menu
* Save your changes

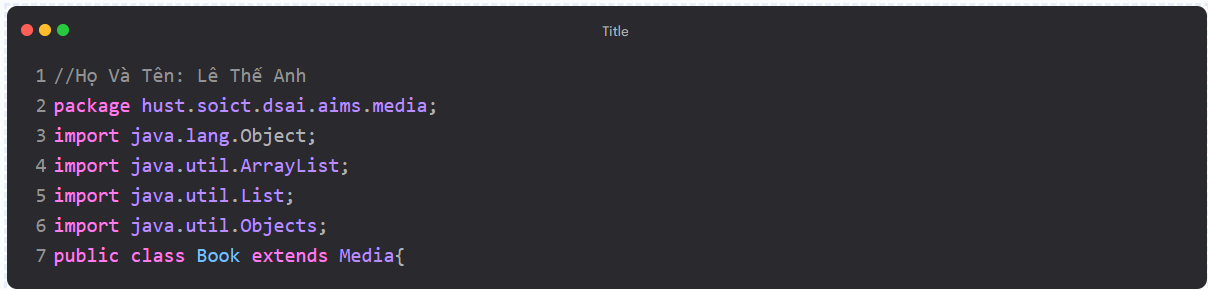


- Do similarly for the **DigitalVideoDisc** class and move it to the package **hust.soict.dsai.aims.media**. Remove the package **hust.soict.dsai.aims.disc.**

* After doing that you will see a lot of errors because of the missing fields
* Extend the **Media** class for both **Book** and **DigitalVideoDisc**
  + **public class Book extends Media**
  + **public class DigitalVideoDisc extends Media**
* Save your changes.

Ảnh có chứa văn bản

Mô tả được tạo tự động



# Creating the **CompactDisc** class

As with **DigitalVideoDisc** and **Book**, the **CompactDisc** class will extend **Media**, inheriting the **id, title**, **category** and **cost** fields and the associated methods.

## 5.1. Create the **Disc** class extending the **Media** class

- The **Disc** class has two fields: **length** and **director**

Ảnh có chứa văn bản

Mô tả được tạo tự động

- Create **getter** methods for these fields

Ảnh có chứa văn bản

Mô tả được tạo tự động

- Create constructor(s) for this class. Use super() if possible.

Ảnh có chứa văn bản

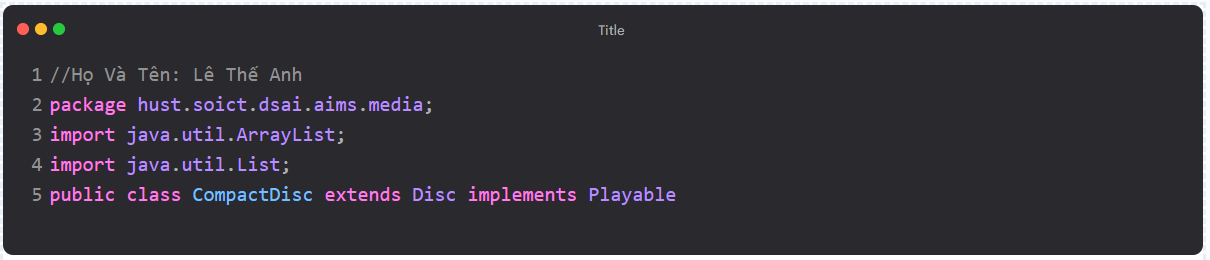
Mô tả được tạo tự động

- Make the **DigitalVideoDisc** extending the **Disc** class. Make changes if need be.

Ảnh có chứa văn bản

Mô tả được tạo tự động

- Create the **CompactDisc** extending the **Disc** class. Save your changes.



## 5.2. Create the **Track** class which models a track on a compact disc and will store information incuding the **title** and **length** of the track

- Add two fields: **String** **title** and **int** **length**

- Make these fields **private** and create their **getter** methods as **public**

- Create constructor(s) for this class.

- Save your changes

Ảnh có chứa văn bản

Mô tả được tạo tự động

## 5.3. Open the **CompactDisc** class

- Add 2 fields to this class:

* a **String** as **artist**
* an **ArrayList** of **Track** as **tracks**

- Make all these fields as **private.** Create public **getter** method for only **artist**.

- Create constructor(s) for this class. Use super() if possible.

- Create methods **addTrack**() and **removeTrack**()

* The **addTrack**() method should check if the input track is already in the list of tracks and inform users
* The **removeTrack**() method should check if the input track existed in the list of tracks and inform users

Ảnh có chứa văn bản

Mô tả được tạo tự động

- Create the **getLength**() method

* Because each track in the CD has a length, the length of the CD should be the sum of lengths of all its tracks.

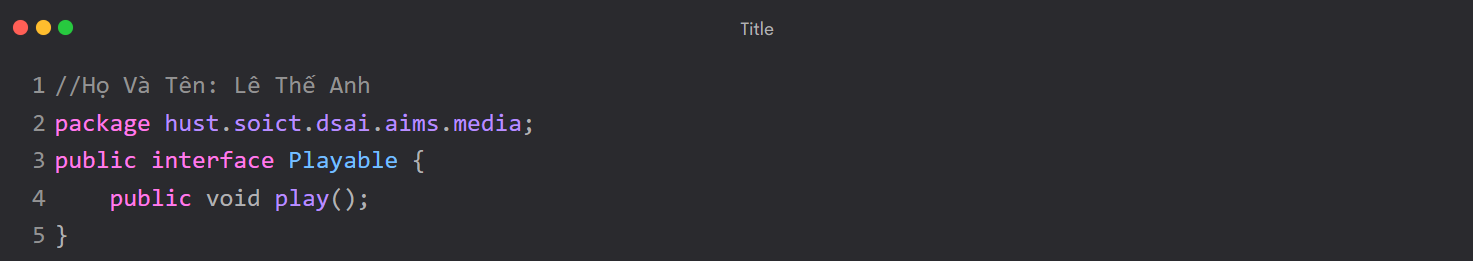
Ảnh có chứa văn bản

Mô tả được tạo tự động

# Create the Playable interface

The **Playable** interface is created to allow classes to indicate that they implement a **play**() method. You can apply Release Flow here by creating a **topic** branch for implementing **Playable** interface.

- Create **Playable** interface, and add to it the method prototype: **public void play();**

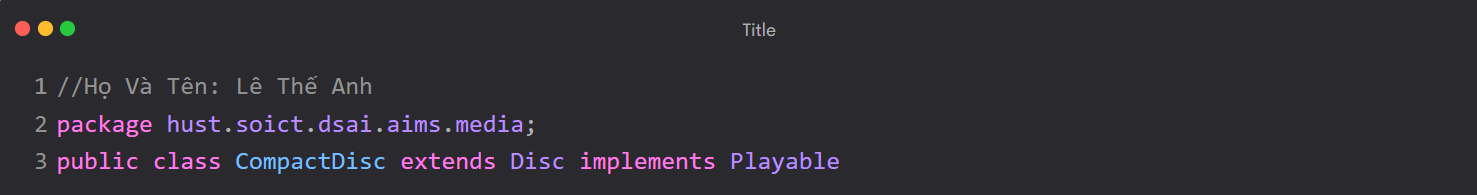
****

- Implement the **Playable** with **CompactDisc**, **DigitalVideoDisc** and **Track**

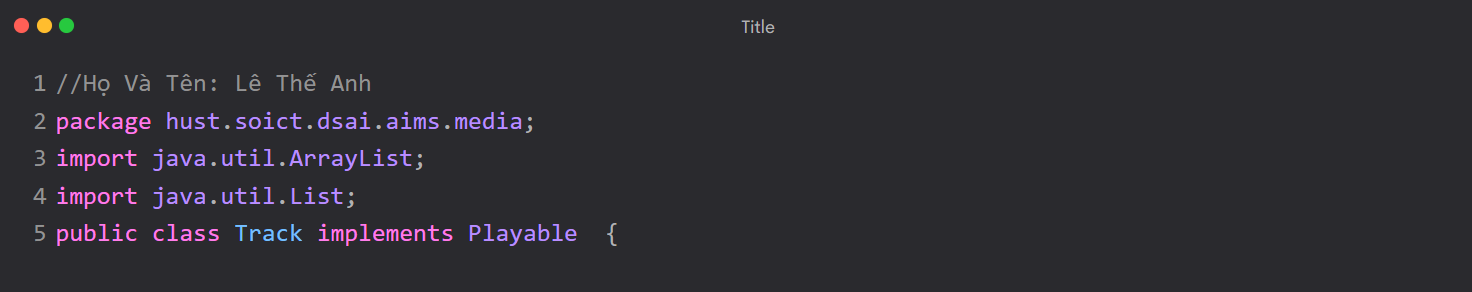
* For each of these classes **CompactDisc** and **DigitalVideoDisc**, edit the class description to include the keywords **implements** **Playable**, after the keyword **extends** **Disc**

Ảnh có chứa văn bản

Mô tả được tạo tự động



* For the **Track** class, insert the keywords **implements** **Playable** after the keywords **public** **class** **Track**



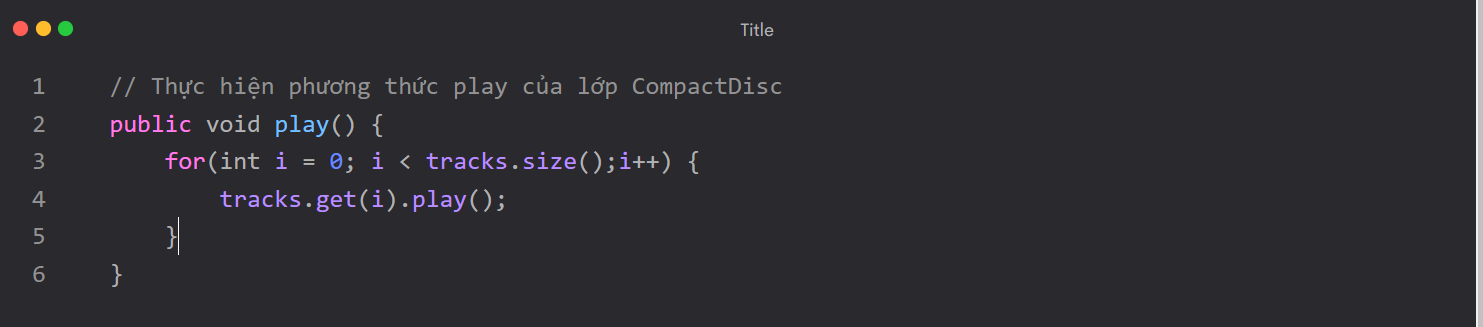
- Implement **play**() for **DigitalVideoDisc** and **Track**

* Add the method **play**() to these two classes
* In the **DigitalVideoDisc**, simply print to screen:

Ảnh có chứa văn bản

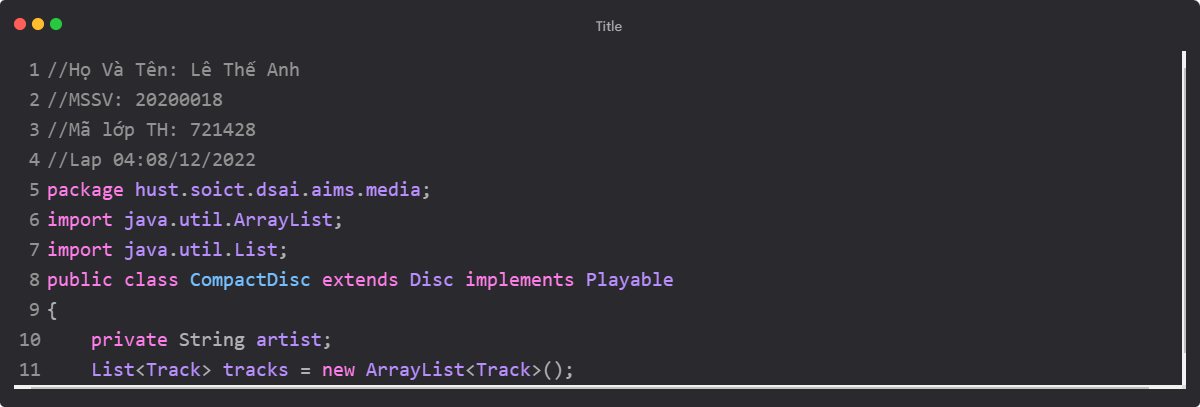
Mô tả được tạo tự động

* Similar additions with the **Track** class



- Implement **play**() for **CompactDisc**

* Since the **CompactDisc** class contains a **ArrayList** of **Tracks**, each of which can be played on its own. The **play**() method should output some information about the **CompactDisc** to console



* Loop through each track of the arraylist and call **Track's** play() method

Ảnh có chứa văn bản

Mô tả được tạo tự động

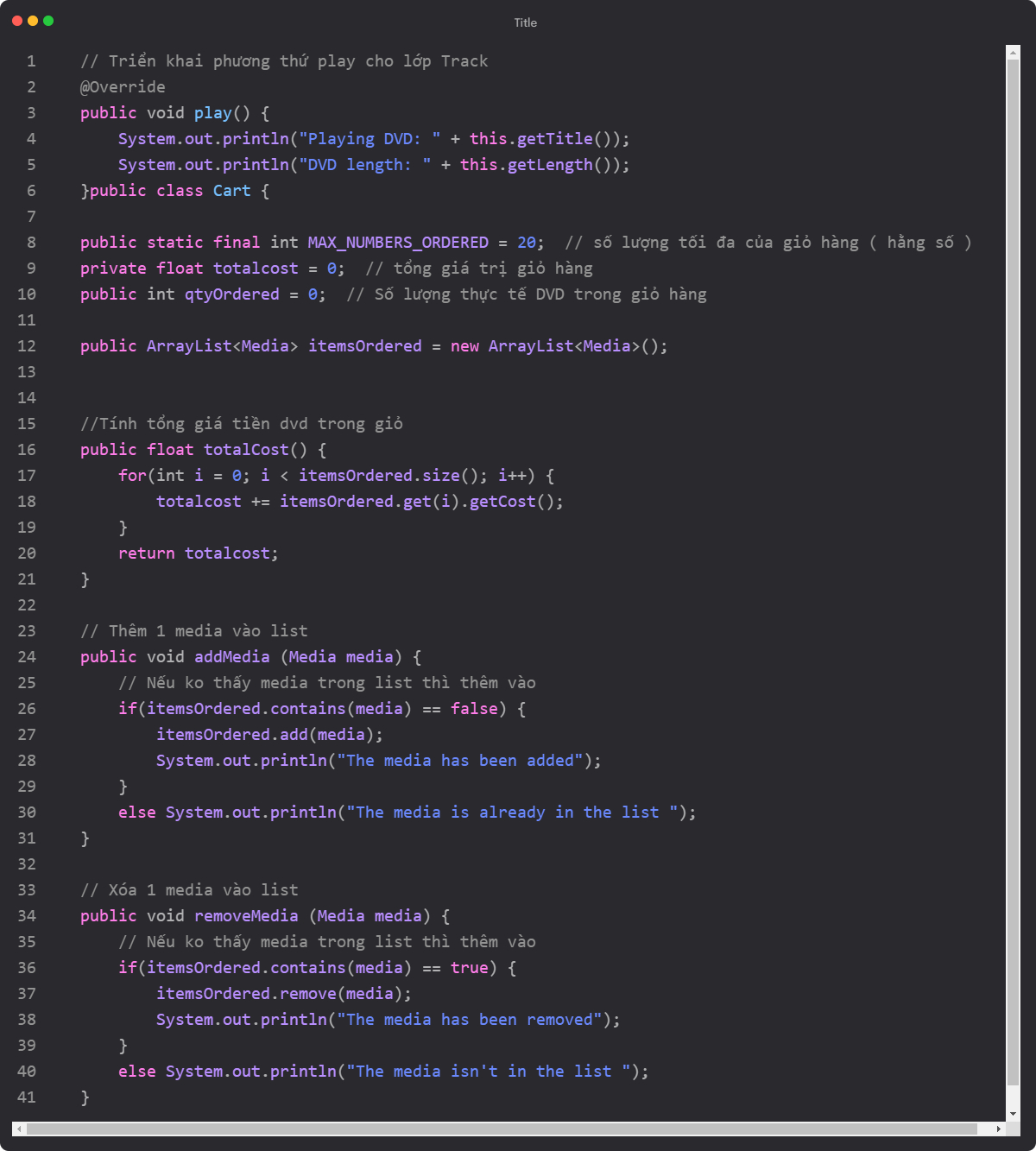
# Update the **Cart** class to work with **Media**

You must now update the **Cart** class to accept not only **DigitalVideoDisc** but also **Book** and **CompactDisc**. Currently, the **Cart** class has methods:

* **addDigitalVideoDisc**()
* **removeDigitalVideoDisc**().

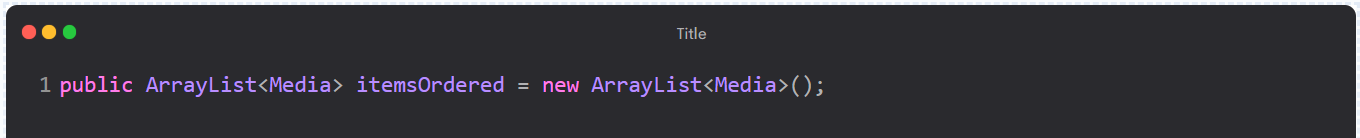
You could add more methods to add and remove **Book** and **CompactDisc**, but since **DigitalVideoDisc**, **Book** and **CompactDisc** are all subclasses of type **Media**, you can simply change **Cart** to maintain a collection of **Media** objects. Thus, you can add a **DigitalVideoDisc**, or a **Book**, or a **CompactDisc** using the same methods.

* Create **addMedia**() and **removeMedia**() to replace **addDigitalVideoDisc**() and **removeDigitalVideoDisc**()
* Update the **totalCost**() method

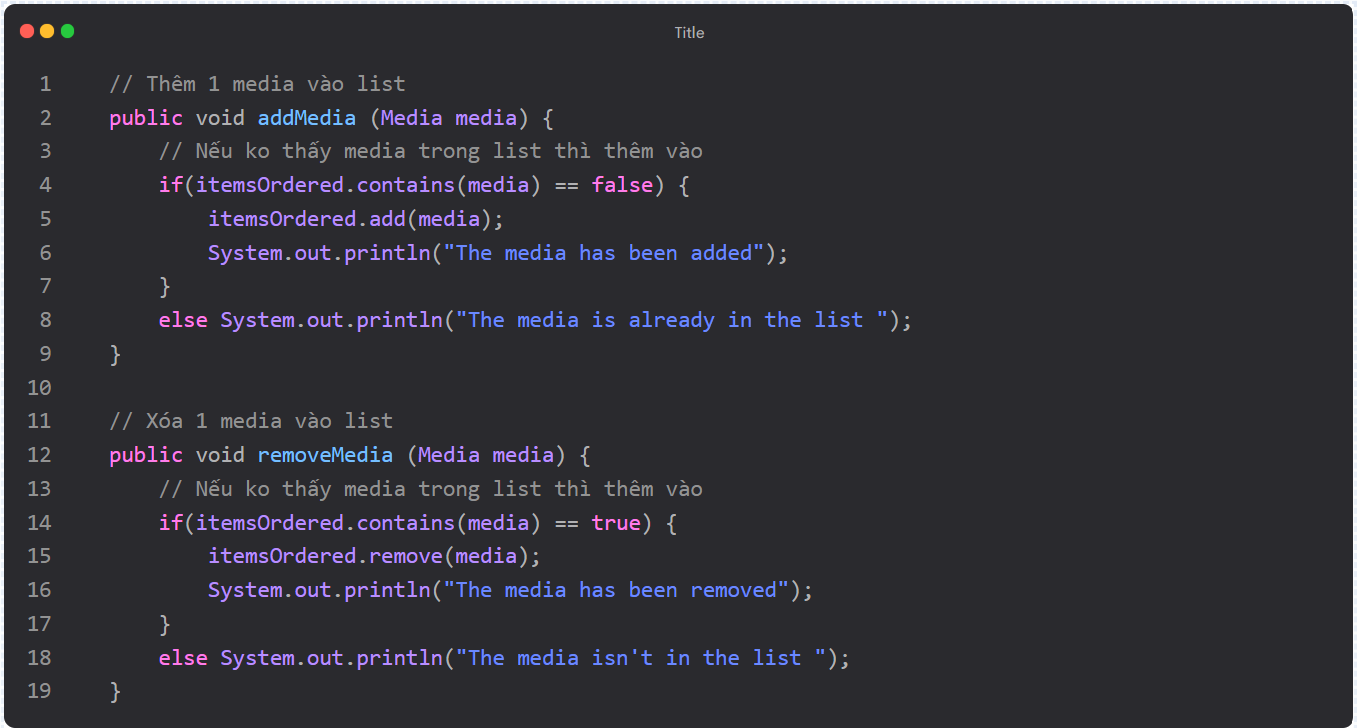


# Update the **Store** class to work with **Media**

* Similar to the **Cart** class, change the **itemsInStore[]** attribute of the **Store** class to **ArrayList<Media>** type.

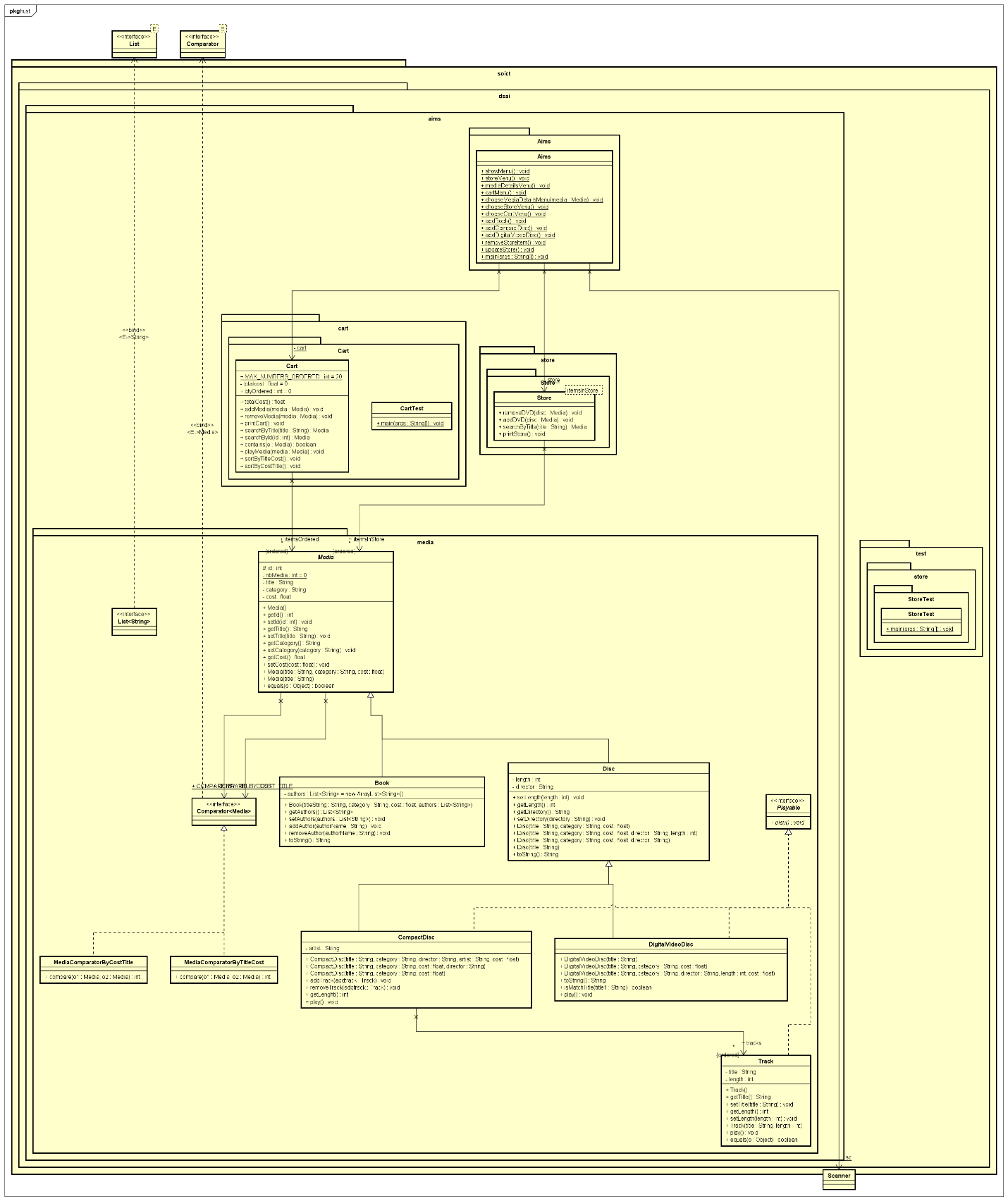


* Replace the **addDigitalVideoDisc**() and **removeDigitalVideoDisc**() methods with **addMedia**() and **removeMedia**()



# Constructors of whole classes and parent classes

- Update the UML class diagram for the **AimsProject**. Update the new .astah & .png file in the **Design** directory. We can apply Release Flow here by creating a branch, e.g., **topic/update-class-diagram/aims-project/lab04,** push the diagram and its image, and then merge with master.

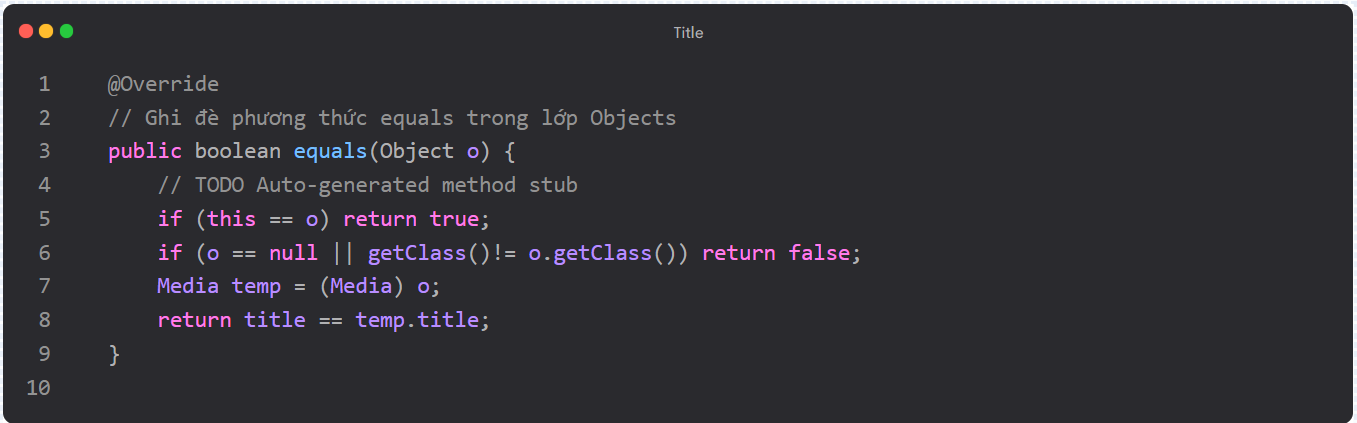


# Unique item in a list

To make sure the list of media in cart or list of tracks in a CD should not contain identical objects, we can override the **equals()** method of the **Object** class

* Please override the boolean equals(Object o) of the Media and the Track class so that two objects of these classes can be considered as equal if:

+ For the Media class: the title is equal

+ For the Track class: the title and the length are equal

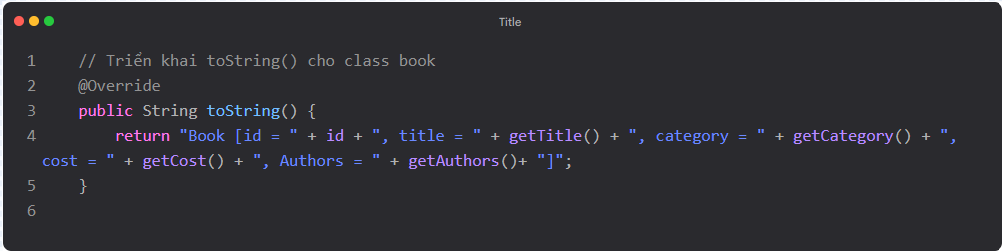
Ảnh có chứa văn bản

Mô tả được tạo tự động

# Polymorphism with toString() method

This exercise gives an illustration for polymorphism at behavior level.

Recall that for each type of media, we have implemented a toString() method that prints out the information of the object. When calling this method, depending on the type of object, corresponding toString() will be performed.



- Create an ArrayList of Media, then add some media (CD, DVD or Book) into the list.

- Iterate through the list and print out the information of the media by using toString() method. Observe what happens and explain in detail.



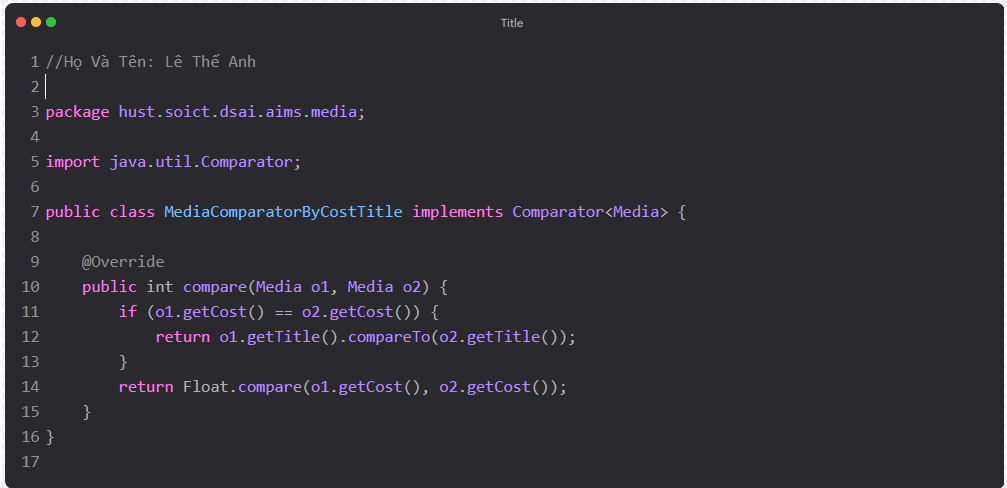
Figure 3. Polymorphism sample code

# Sort media in the cart

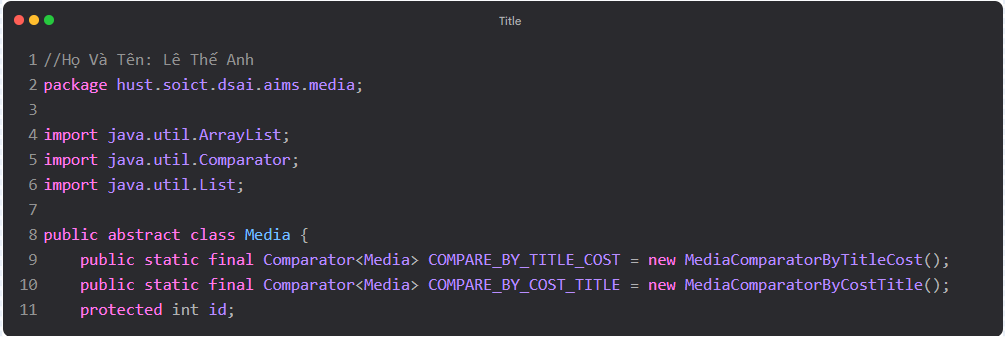
As mentioned before, when seeing the current cart, the user can sort the items in the cart by title or by cost:

* Sort by title: the system displays all the medias in the alphabet sequence by title. In case they have the same title, the medias having the higher cost will be displayed first.
* Sort by cost: the system the system displays all the medias in decreasing cost order. In case they have the same cost, the medias will be ordered by title (alphabetical).
* Create two classes of comparators, one for each type of ordering





* Implement the compare() method of each comparator class to reflect the ordering that we want, either by title then cost, or by cost then title. You may utilize the method **Comparator.thenComparing()** to sort using multiple fields.
* Add the comparators as attributes of the Media class:



* Pass the comparator into Collections.sort:

Ảnh có chứa văn bản

Mô tả được tạo tự động

**Question**: Alternatively, to compare items in the cart, instead of using Comparator, we can use the Comparable interface and override the compareTo()method. You can refer to the Java docs to see the information of this interface.

Suppose we are taking this Comparable interface approach.

* What class should implement the Comparable interface?
* In those classes, how should you implement the compareTo()method be to reflect the ordering that we want?
* Can we have two ordering rules of the item (by title then cost and by cost then title) if we use this Comparable interface approach?
* Suppose the DVDs has a different ordering rule from the other media types, that is by title, then decreasing length, then cost. How would you modify your code to allow this?

# Create a complete console application in the Aims class

In the **main** method of **Aims**, you will now implement a complete console application, by first create an instance of the **Store** class and then, provide a list of functionalities through a menu that the user can interact with. For the home interface, you will create the main menu as following:



* From the main menu, if the user choses option “**View store**”, the application will display all the items in the store, and a menu as following:

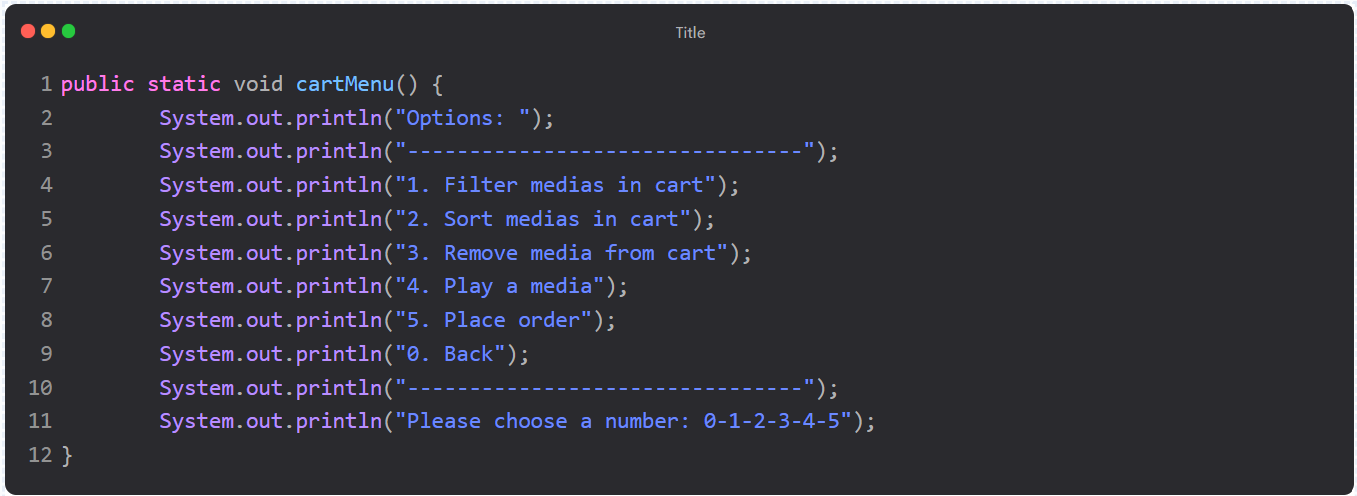


* The option “**See a media’s details**” will ask the user to enter the title of the media and display the information of that media. Please remember to check the validity of the title. Under the information display, the system also shows the following menu (note that the “**Play**” option is only available to CD and DVD type.

Ảnh có chứa văn bản

Mô tả được tạo tự động

* The option “**Add a media to cart**” will ask the user to enter the title of the media that he/she sees on the screen (the list of medias in store), then add the media to cart. Please remember to check the validity of the title. After adding a DVD to cart, the system will display the number of DVDs in the current cart.
* The option “**Play a media**” will ask the same input from the user as option 2. You should again check the validity of the title.
* From the main menu, if the user choses option “**Update store**”, the application will allow the user to add a media to or remove a media from the store
* From the main menu, if the user choses option “**See current cart**”, the application will display the information of the current cart, and a menu as following:



The “**Filter medias in cart”** option should allow the user to choose one of two filtering options: by id and by title.

The “**Sort medias in cart**” option should allow the user to choose one of two sorting option: by title or by cost.

Ảnh có chứa văn bản

Mô tả được tạo tự động

**Note**: When the user chooses option “**Place order**”, the system is supposed to move on to the Delivery Information gathering & Payment step. However, for simplicity, within the scope of this lab course, when the user chooses this option, we only need to notify the user that an order is created and empty the current cart.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235  236  237  238  239  240  241  242  243  244  245  246  247  248  249  250  251  252  253  254  255  256  257  258  259  260  261  262  263  264  265  266  267  268  269  270  271 | **private** **static** **final** Store store = **new** Store();  **private** **static** **final** Cart cart = **new** Cart();  *// Chọn MediaDetailsMenu*  **public** **static** **void** chooseMediaDetailsMenu(Media media) {  **while**(**true**) {  mediaDetailsMenu();  **int** choose = sc.nextInt();  **switch** (choose) {  **case** 1:  cart.addMedia(media);  **break**;  **case** 2:  cart.playMedia(media);  **break**;  **case** 0:  **return**;  }  }  }  *// Chọn StoreMenu()*  **public** **static** **void** chooseStoreMenu() {  **while** (**true**) {  storeMenu();  **int** choose = sc.nextInt();  **switch** (choose) {  **case** 1: {*// Tìm media*  System.out.println("Enter media title:");  sc.nextLine();  String title = sc.nextLine();  Media m = store.searchByTitle(title);  **if** (m == **null**) {  System.out.println("No media with title: " + title + " found");  } **else** {  System.out.println("Found:");  System.out.println(m);  chooseMediaDetailsMenu(m);  }  }  **break**;  **case** 2: *// add Media vào card*  {  System.out.println("Enter media title to add to cart:");  String title = sc.nextLine();  Media m = store.searchByTitle(title);  **if** (m == **null**) {  System.out.println("No media with title: " + title + " found");  } **else** {  System.out.println("Found:");  System.out.println(m);  cart.addMedia(m);  System.out.println("Number of media in the cart: " + cart.itemsOrdered.size());  }  }  **break**;  **case** 3: //*tìm kiếm Media*  {  System.out.println("Enter media title to add to cart:");  String title = sc.nextLine();  Media m = store.searchByTitle(title);  **if** (m == **null**) {  System.out.println("No media with title: " + title + " found");  } **else** {  cart.playMedia(m);  }  }  **break**;  **case** 0:  **return**;  }  }  }  *// Chọn Cart Menu*  **public** **static** **void** chooseCartMenu() {  **while** (**true**) {  cartMenu();  **int** choose = sc.nextInt();  **switch** (choose) {  **case** 1:  {  System.out.println("Filter media by id --> enter 0");  System.out.println("Filter media by title --> enter 1");  **int** c = sc.nextInt();  **if** (c == 0) {  System.out.println("Enter id:");  **int** id = sc.nextInt();  Media m = cart.searchById(id);  **if** (m == **null**) {  System.out.println("No item with id: " + id + " found");  } **else** {  System.out.println("Found item with id: " + id);  System.out.println(m);  }  } **else** {  sc.nextLine();  System.out.println("Enter title:");  String title = sc.nextLine();  Media m = cart.searchByTitle(title);  **if** (m == **null**) {  System.out.println("No item with title: " + title + " found");  } **else** {  System.out.println("Found item with title: " + title);  System.out.println(m);  }  }  }  **break**;  **case** 2:  {  System.out.println("Sort by title --> enter 0");  System.out.println("Sort by cost --> enter 1");  **int** c = sc.nextInt();  **if** (c == 0) {  cart.sortByTitleCost();  } **else** {  cart.sortByCostTitle();  }  }  **break**;  **case** 3:  {  System.out.println("Enter media title to remove");  String title = sc.nextLine();  Media m = cart.searchByTitle(title);  **if** (m == **null**) {  System.out.println("No item with title: " + title + " found");  } **else** {  cart.removeMedia(m);  }  }  **break**;  **case** 4:  {  System.out.println("Enter media title to play");  String title = sc.nextLine();  Media m = cart.searchByTitle(title);  **if** (m == **null**) {  System.out.println("No item with title: " + title + " found");  } **else** {  cart.playMedia(m);  }  }  **break**;  **case** 5:  {  System.out.println("Your order is created");  System.out.println("Your cart now is empty");  }  **break**;  **case** 0:  **return**;  }  }  }  *// Thêm Book vào Store*  **public** **static** **void** addBook() {  System.out.println("Enter book title, category, cost, number of authors");  **int** n;  sc.nextLine();  String title = sc.nextLine();  String category = sc.nextLine();  Float cost = sc.nextFloat();  n = sc.nextInt();  sc.nextLine();  List<String> authors = **new** ArrayList<>();  Book book = **new** Book(title, category, cost, authors);  System.out.println("Enter " + n + " author's name:");  **for** (**int** i = 1; i <= n; i++) {  String author = sc.nextLine();  System.out.println(author);  book.addAuthor(author);  }  store.addDVD(book);  }  *// Thêm đĩa CD vào Store*  **public** **static** **void** addCompactDisc() {  System.out.println("Enter compact disc title, category, cost, director, artist, number of tracks");  **int** n;  sc.nextLine();  String title = sc.nextLine();  String category = sc.nextLine();  Float cost = sc.nextFloat();  sc.nextLine();  String director = sc.nextLine();  String artist = sc.nextLine();  n = sc.nextInt();  List<Track> tracks = **new** ArrayList<>();  CompactDisc compactDisc = **new** CompactDisc(title, category, director, artist, cost);  System.out.println("Enter " + n + " tracks info");  **for** (**int** i = 1; i <= n; i++) {  System.out.println("Enter track " + i + " title and length:");  String trackTitle = sc.nextLine();  **int** length = sc.nextInt();  compactDisc.addTrack(**new** Track(trackTitle, length));  }  store.addDVD(compactDisc);  }  *// Thêm 1 dvd vào store*  **public** **static** **void** addDigitalVideoDisc() {  System.out.println("Enter digital video disc title, category, cost, director, length");  **int** n;  String title = sc.nextLine();  String category = sc.nextLine();  Float cost = sc.nextFloat();  String director = sc.nextLine();  **int** length = sc.nextInt();  DigitalVideoDisc disc = **new** DigitalVideoDisc(title, category, director, length, cost);  store.addDVD(disc);  }  *// Xóa 1 items trong store*  **public** **static** **void** removeStoreItem() {  System.out.println("Enter remove item title");  String title = sc.nextLine();  Media media = store.searchByTitle(title);  **if** (media != **null**) {  store.removeDVD(media);  }  }  *// Update Store*  **public** **static** **void** updateStore() {  System.out.println("Enter your choice number");  System.out.println("1. Add a book");  System.out.println("2. Add a compactDisc");  System.out.println("3. Add a digitalVideoDisc");  System.out.println("4. Remove an item");  **int** choose = sc.nextInt();  **switch** (choose) {  **case** 1:  addBook();  **break**;  **case** 2:  addCompactDisc();  **break**;  **case** 3:  addDigitalVideoDisc();  **break**;  **case** 4:  removeStoreItem();  **break**;  }  }  **public** **static** **void** main(String[] args) {  sc = **new** Scanner(System.in);  **while** (**true**) {  showMenu();  **int** choose = sc.nextInt();  **switch** (choose) {  **case** 1:  store.printStore();  chooseStoreMenu();  **break**;  **case** 2:  updateStore();  **break**;  **case** 3:  cart.printCart();  chooseCartMenu();  **break**;  **case** 0:  System.out.println("Bye bye");  **return**;  }  }  }  } |